This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Original) An output engine operable to convert a file from a first format to a second format, the output engine comprising:

a decomposer operable to be called by a calling application and to receive a file and a desired file format from the calling application, the decomposer operable to decompose the file into a component architecture; and

a writer operable to call the decomposer multiple times to retrieve the component architecture of the file and generate a new version of the file in the second format.

2. (Original) An output engine as claimed in claim 1, further comprising a second writer operable to call the decomposer multiple times to retrieve the component architecture of the file and generate a new version of the file in a third format.

3. (Original) An output engine as claimed in claim 1, wherein the decomposer includes a job processor.

4. (Original) An output engine as claimed in claim 3, wherein the job processor is operable to load pages of the file and associate data with each page.

5. (Original) An output engine as claimed in claim 3, wherein the decomposer includes a writer interface.

6. (Original) An output engine as claimed in claim 5, wherein the decomposer includes a calling application interface.

7. (Original) An output engine as claimed in claim 3, wherein the writer includes a job processor interface.

8. (Currently Amended) An output engine ~~as claimed in claim 7,~~ operable to convert a file from a first format to a second format, the output engine comprising:

a decomposer operable to be called by a calling application and to receive a file and a desired file format from the calling application, the decomposer operable to decompose the file into a component architecture and make the component architecture available to a writer, the component architecture including objects included in the file; and

a writer operable to call the decomposer multiple times to retrieve the objects included in the file and generate a new version of the file in the second format, wherein the writer includes a layer processor, a collection processor, and an item processor.

9. (Original) An output engine as claimed in claim 8, wherein the writer includes a stack and an output module.

10. (Currently Amended) A method of converting an input file from a first format to a second format, the method comprising:

delivering a desired file format and the input file to a decomposer;

decomposing the input file into a component architecture in the decomposer, the component architecture including properties of objects included in the input file; ~~and~~

making the component architecture available to a writer; and

generating a new version of the input file in the second format by calling the decomposer multiple times from a writer to obtain the properties of the objects included in the input file.

11. (Original) A method of converting an input file as claimed in claim 10, the method further comprising:

sending the new version of the input file to a second writer; and

generating a new version of the input file in a third format.

12. (Original) A method of converting an input file as claimed in claim 10, the method further comprising converting the input file from a fourth format to a common file format prior to delivering the input file to a decomposer.

13. (Currently Amended) A method of converting an input file having ~~at plurality~~ a plurality of pages and formatted in a first format to an output file formatted in a second format, the method comprising:

receiving a file conversion request from a calling application;

loading each page of the input file, one page at a time, in a decomposer;

associating data with one or more of the plurality of pages;

decomposing objects in each page into a component architecture, the component architecture including properties of objects included in the input file;

making the component architecture available to a writer;

driving each page to a writer; and

generating the output file by calling a decomposer multiple times from a writer in order to obtain the properties of objects included in the input file.

14. (Original) A method as claimed in claim 13, further comprising:

executing a plurality of writers in a chained fashion.

15. (Original) A method as claimed in claim 13, further comprising

sending the output file to a second writer; and

generating a new version of the output file in a third format.

16. (Currently Amended) A method ~~as claimed in claim 13, further comprising~~ of converting an input file having a plurality of pages and formatted in a first format to an output file formatted in a second format, the method comprising:

receiving a file conversion request from a calling application;

loading each page of the input file, one page at a time, in a decomposer;

associating data with one or more of the plurality of pages;

determining layers, collections, and items in each page;

pushing the determined layers, collections, and items onto a stack; ~~and~~

decomposing objects in each page into a component architecture, the component architecture including properties of objects included in the input file;

making the component architecture available to a writer;

driving each page to a writer;

assembling the determined layers, collections, and items into the output file; and

generating the output file by calling a decomposer multiple times from a writer in order to obtain the properties of objects included in the input file.

17. (Currently Amended) A file conversion system comprising:

a workstation having a source application, an output engine, and a document manager, the output engine including

a decomposer operable to be called by a calling application and to receive a file and a desired file format from the calling application, the decomposer operable to decompose the file into a component architecture, the component architecture including properties of objects included in the file; and

a writer operable to call the decomposer multiple times to retrieve the ~~component architecture of~~properties of the objects included in the file and generate a new version of the file in the second format.

a form data database and a form database, each accessible to the workstation; and

a server accessible to the workstation and having a document control and production engine.

18. (Original) A system as claimed in claim 17, wherein the output engine further comprises a second writer operable to call the decomposer multiple times to retrieve the component architecture of the file and generate a new version of the file in a third format.

19. (Original) A system as claimed in claim 17, wherein the decomposer includes a job processor.

20. (Original) A system as claimed in claim 19, wherein the job processor is operable to load pages of the file and associate data with each page.

21. (Original) A system as claimed in claim 19, wherein the decomposer includes a writer interface.

22. (Original) A system as claimed in claim 21, wherein the decomposer includes a calling application interface.

23. (Original) A system as claimed in claim 19, wherein the writer includes a job processor interface.

24. (Currently Amended) A file conversion system ~~system as claimed in claim 17, wherein the writer includes a layer processor, a collection processor, and an item processor.~~comprising:

a workstation having a source application, an output engine, and a document manager, the output engine including

a decomposer operable to be called by a calling application and to receive a file and a desired file format from the calling application, the decomposer operable to decompose the file into a component architecture, the component architecture including objects included in the file; and

a writer operable to call the decomposer multiple times to retrieve the objects included in the file and generate a new version of the file in the second format, the writer including a layer processor, a collection processor, and an item processor.

a form data database and a form database, each accessible to the workstation; and

a server accessible to the workstation and having a document control and production engine.

25. (Currently Amended) A system as claimed in claim 24~~17~~, wherein the writer includes a stack and an output module.

26. (Original) A system as claimed in claim 17, further comprising a converter accessible to the workstation and operable to convert files from a foreign format to a common format.

27. (New) A method of converting an input file from a first format to a second format, the method comprising:

delivering a desired file format and the input file to a decomposer;

decomposing the input file into a component architecture in the decomposer;

generating at least one data file, the at least one data file representing an aspect of a document;

generating an object model for the at least one data file, the object model having one or more objects;

making the object model for the at least one data file available to a writer; and

generating a new version of the input file in the second format by calling the decomposer multiple times from a writer.

28. (New) A method as claimed in claim 27, further comprising:

delivering a writer identifier to the decomposer;

selecting a writer based on the writer identifier; and

making the object model for the at least one data file available to the selected writer.

29. (New) A method as claimed in claim 27, wherein the input file includes a plurality of pages and the method further comprises loading each page of the input file and associating data with each page in a job processor.

30. (New) An output engine operable to convert a file from a first format to a second format, the output engine comprising:

a decomposer operable to be called by a calling application and to receive, from the calling application, a file, a desired file format, and a set of parameters used to set properties of a plurality of writers, the decomposer operable to decompose the file into a component architecture; and

a plurality of writers, each writer having an identifier and configured to be selected according to at least one of the set of parameters used to set properties of a plurality of writers, and to call the decomposer multiple times to retrieve the component architecture of the file and generate a new version of the file in the second format.